Technical Cybersecurity

ret2libc design

Let’s write one!

X86 RET2LIBC SUMMARY

- We’re going to find and call the **system(.)** function in libc
- **system(.)** executes arbitrary commands on the host system
  - We’re going to call /bin/bash, but you could call anything (e.g. socat, netcat, and so on)
- We need to get an argument to **system(.)**
  - We’ll use environment variables
  - Env vars are read into all program’s memory image

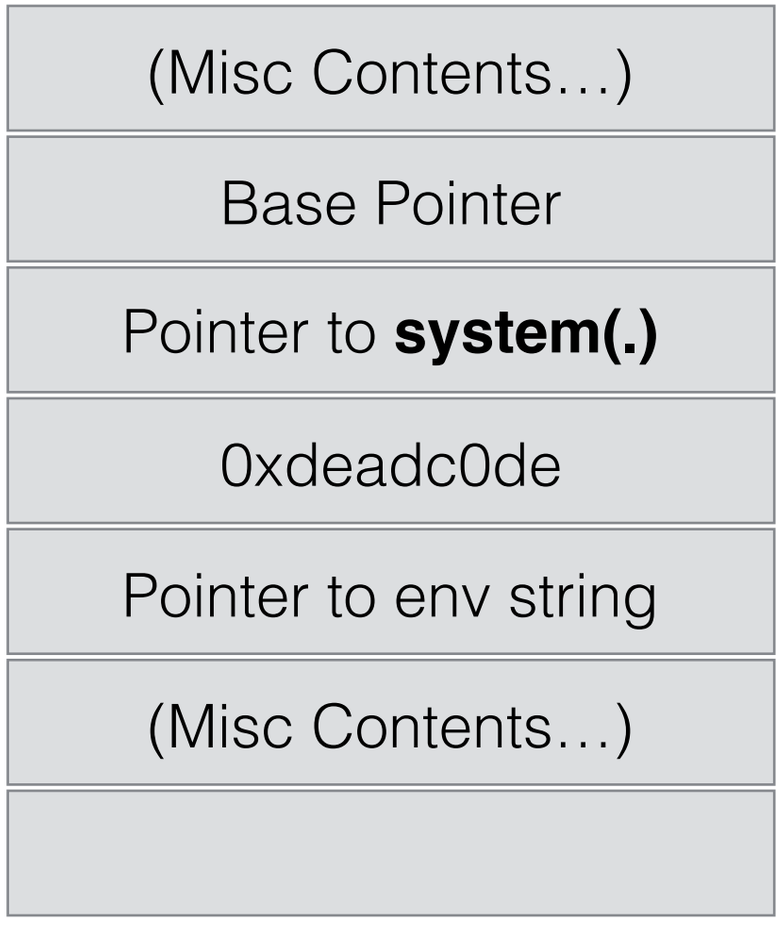
Stack Layout

Pointer to system

- Inserted over the RA pointer

Pointer to string

- Inserted over



getenv.c

```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    if (argc != 2) {
      exit(0);
    }
    char* varname = argv[1];
    char* ptr = getenv(varname);
    printf("%p\n", ptr);
}
```

```
cclamb@ubuntu:~/Work/abi-playground $ export PAYLOAD="/bin/bash"
cclamb@ubuntu:~/Work/abi-playground $ ./getenv PAYLOAD
0xffffdab7
cclamb@ubuntu:~/Work/abi-playground $ ./getenv PAYLOAD
0xffffdab7
cclamb@ubuntu:~/Work/abi-playground $ ./getenv PAYLOAD
0xffffdab7
cclamb@ubuntu:~/Work/abi-playground $
```

```
Reading symbols from smash...done.
(gdb) x/s 0xffffdab7
0xffffdab7:     <error: Cannot access memory at address 0xffffdab7>
(gdb) b main
Breakpoint 1 at 0x804846e: file smash.c, line 11.
(gdb) r
Starting program: /home/cclamb/Work/abi-playground/smash

Breakpoint 1, main (argc=1, argv=0xffffcef4) at smash.c:11
11          char* arg = argv[1];
(gdb) x/s 0xffffdab7
0xffffdab7:     "ODULES=gail:atk-bridge"
(gdb) x/20s 0xffffdab7
0xffffdab7:     "ODULES=gail:atk-bridge"
0xffffdace:     "COLUMNS=103"
0xffffdada:     "WINDOWPATH=2"
0xffffdae7:     "PAYLOAD=/bin/bash"
0xffffdaf9:     "VIRTUALENVWRAPPER_SCRIPT=/usr/local/bin/virtualenvwrapper.sh"
0xffffdb36:     "SHELL=/bin/bash"
0xffffdb46:     "TERM=screen"
0xffffdb52:     "VTE_VERSION=5202"
0xffffdb63:     "QT_IM_MODULE=ibus"
0xffffdb75:     "XMODIFIERS=@im=ibus"
0xffffdb89:     "IM_CONFIG_PHASE=2"
0xffffdb9b:     "XDG_CURRENT_DESKTOP=ubuntu:GNOME"
0xffffdbbc:     "GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1"
0xffffdbf0:     "POWERLINE_COMMAND=powerline"
0xffffdc0c:     "GNOME_TERMINAL_SERVICE=:1.325"
0xffffdc2a:     "ROPEME_HOME=/home/cclamb/Work/ropeme"
0xffffdc4f:     "TMUX_PANE=%0"
0xffffdc5c:     "SHLVL=2"
0xffffdc64:     "XDG_SEAT=seat0"
0xffffdc73:     "GDMSESSION=ubuntu"
(gdb) x/s 0xffffdaed
0xffffdaed:     "D=/bin/bash"
(gdb) x/s 0xffffdaef
0xffffdaef:     "/bin/bash"
(gdb) p system
$1 = {<text variable, no debug info>} 0xf7e1bd10 <system>
```

We have some addresses!

System

- **(gdb) p system** gives us 0xf7e1bd10
  - p prints information associated with a symbol, here, the **system(.)** function
  - our **getenv** program gave us a starting point to find our PAYLOAD environment variable, 0xffffdaef
    - We only care about the string, not the environment variable name

With no ASLR…

- …the **system(.)** address will not change
- …the PAYLOAD address **WILL** still shift a bit
  - We'll need to hunt for it outside of the debugger when we get to that point

Okay, let’s start building.